Learn to Build an End-to-End Machine Learning Pipeline - Part 2
Project Overview

Overview

The project addresses a critical challenge faced by the logistics industry. Delayed truck shipments not only result in increased operational costs but also impact customer satisfaction. Timely delivery of goods is essential to meet customer expectations and maintain the competitiveness of logistics companies.

By accurately predicting truck delays, logistics companies can:

- Improve operational efficiency by allocating resources more effectively
- Enhance customer satisfaction by providing more reliable delivery schedules
- Optimize route planning to reduce delays caused by traffic or adverse weather conditions
- Reduce costs associated with delayed shipments, such as penalties or compensation to customers

In the first phase of our three-part series, **Learn to Build an End-to-End Machine Learning Pipeline - Part 1**, we laid the groundwork by utilizing PostgreSQL and MySQL in AWS RDS for data storage, setting up an AWS Sagemaker Notebook, performing data retrieval, conducting exploratory data analysis, and creating feature groups with Hopsworks.

In Part 2, we delve deeper into the machine-learning pipeline. Focusing on data retrieval from the feature store, train-validation-test split, one-hot encoding, scaling numerical features, and leveraging Weights and Biases for model experimentation, we will build our pipeline for model building with logistic regression, random forest, and XGBoost models. Further, we explore hyperparameter tuning with sweeps, discuss grid and random search, and, ultimately, the deployment of a Streamlit application on AWS.

Note: AWS Usage Charges

This project leverages the AWS cloud platform to build the end-to-end machine learning pipeline. While using AWS services, it's important to note that certain activities may incur charges. We recommend exploring the AWS Free Tier, which provides limited access to a wide range of AWS services for 12 months. Please refer to the **AWS Free Tier page** for detailed information, including eligible services and usage limitations.

Aim

The project aims to develop an end-to-end machine learning pipeline leveraging Hopsworks' feature store and model experimentation with Weights and Biases, ultimately deploying a Streamlit application on AWS.

Data Description

The project involves the following data tables:

- City Weather: Weather data for various cities
- Routes: Information about truck routes, including origin, destination, distance, and travel time
- Drivers: Details about truck drivers, including names and experience
- Routes Weather: Weather conditions specific to each route
- Trucks: Information about the trucks used in logistics operations
- Traffic: Traffic-related data
- Truck Schedule: Schedules and timing information for trucks

Tech Stack

➔ Language: Python 3.10
➔ Libraries: NumPy, Pandas
➔ Data: Hopsworks Feature Store
➔ Experiment Tracking: Weights and Biases
➔ Model Building: Scikit-learn, XGBoost
➔ Cloud Platform: AWS Sagemaker, AWS EC2

Approach

- Data Retrieval from Hopsworks:
    - Connecting Hopsworks with Python.
    - Retrieving data directly from the feature store.

- Train-Validation-Test Split

- One-Hot Encoding

- Scaling Numerical Features

- Model Experimentation and Tracking:

    - Weights and Biases Introduction
    - Setting up a new project and connecting it to Python

- Model Building
    - Logistic Regression
    - Random Forest
    - XGBoost

- Hyperparameter Tuning with Sweeps

- Streamlit Application Development and Fetching the Best Model

- Deployment on AWS EC2 Instance

Code Folder Overview:

Once you unzip the code.zip file, you can find the following folders:

```
├─ Data
│   ├─ Database_backup
│   │   ├─ truck-eta-mysql.sql
│   │   └─ truck-eta-postgres.sql
│   ├─ data_description.pdf
│   └─ Training_data
│       ├─ city_weather.csv
│       ├─ drivers_table.csv
│       ├─ routes_table.csv
│       ├─ routes_weather.csv
│       ├─ traffic_table.csv
│       ├─ trucks_table.csv
│       └─ truck_schedule_table.csv
├─ Deployment
│   ├─ app.py
│   ├─ app_config.yaml
│   ├─ requirements.txt
│   ├─ truck_data_encoder.pkl
│   └─ truck_data_scaler.pkl
├─ Notebooks
│   ├─ Truck-Delay-Classification_Part_2.ipynb
│   └─ Truck_Delay_Classification_Part_1.ipynb
└─ References
    └─ readme.md
    |_ solution_methodology.pdf
```

Here is a brief information on the folders:

1. Data
2. Notebook
3. Deployment
4. References

1. The data folder contains database backup files in both MySQL and PostgreSQL formats. These files can be used to restore a database. It also contains training data files in CSV format and data descriptions.

2. The notebook folder contains the original ipython notebooks as in the lectures. Note that this notebook contains code, observation, and other information per videos.

3. The reference folder contains a README.md file, which includes documentation instructions, and solution methodology of the project.

4. The deployment folder contains files essential for deploying the Streamlit application, including the application script (app.py), configuration settings (app_config.yaml), required dependencies (requirements.txt), and pre-trained encoders and scalers for truck data.

Project Takeaways

1. How to connect Python with Hopsworks and fetch data?
2. Understand the significance of train validation test data splitting
3. Implement one-hot encoding for categorical variables.
4. Distinguish between fit-transform and transform, storing for future use.
5. Implement normalization techniques in Python.
6. Understand the significance of experiment tracking.
7. How to connect with Weights and Biases for model experimentation?
8. Implement and Track Logistic regression, Random forest, and XGBoost models.
9. Explore model evaluation metrics and their business implications.
10. Utilize hyperparameter sweeps in Weights and Biases for tuning.
11. Learn to fetch the best model from Weights and Biases
12. Develop a Streamlit application and deploy it on AWS